

Lecture 16: Reinforcement Learning

CS486/686 Intro to Artificial Intelligence

2026-3-5

Pascal Poupart
David R. Cheriton School of Computer Science
CIFAR AI Chair at Vector Institute



Outline

- Reinforcement Learning
 - Q-Learning
 - Exploration strategies

What is Reinforcement Learning?

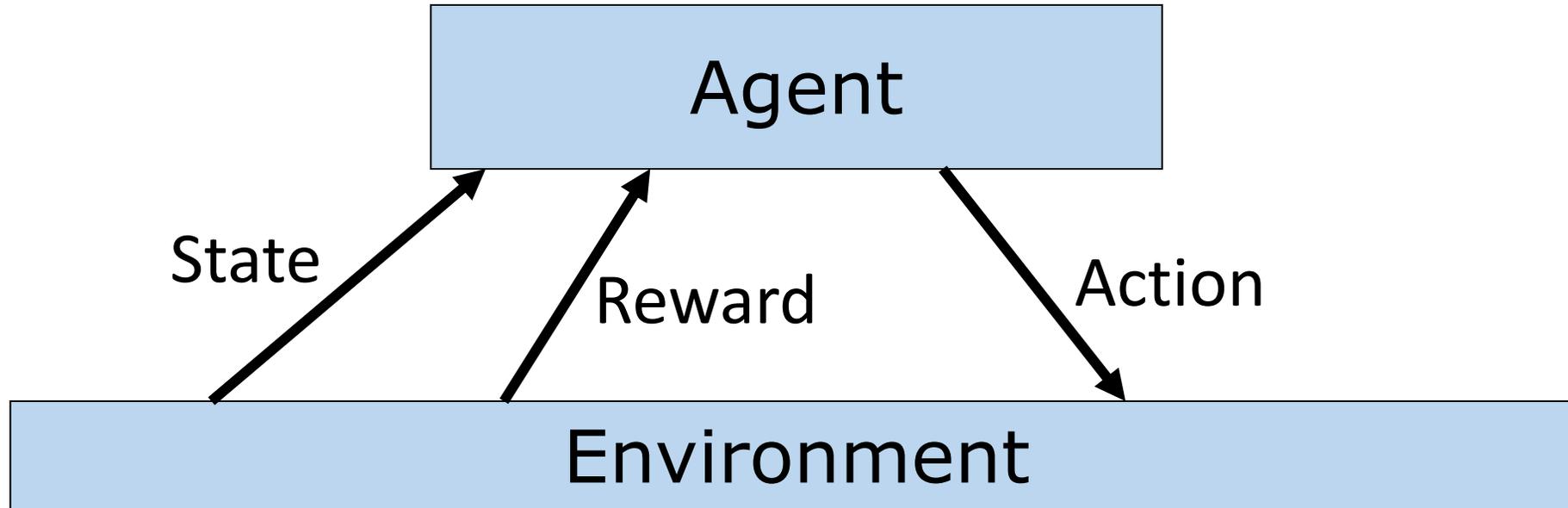
- Reinforcement learning is also known as
 - Optimal control
 - Approximate dynamic programming
 - Neuro-dynamic programming
- [Wikipedia](#): reinforcement learning is an area of machine learning inspired by behavioural psychology, concerned with how software **agents** ought to take **actions** in an **environment** so as to maximize some notion of cumulative **reward**.

Animal Psychology

- Negative reinforcements
 - Pain and hunger
- Positive reinforcements
 - Pleasure and food
- Reinforcements used to train animals
- Let's do the same with computers



Reinforcement Problem



Goal: Learn to choose actions that maximize rewards

Reinforcement Learning

- Comprehensive, but challenging form of machine learning
 - Stochastic environment
 - Incomplete model
 - Interdependent sequence of decisions
 - No supervision
 - Partial and delayed feedback
- Long term goal: **continual learning**

Reinforcement Learning

- Formal Definition

- States: $s \in S$

- Actions: $a \in A$

- Rewards: $r \in \mathbb{R}$

- ~~▪ Transition model: $\Pr(s_t | s_{t-1}, a_{t-1})$~~

- ~~▪ Reward model: $\Pr(r_t | s_t, a_t)$~~

- Discount factor: $0 \leq \gamma \leq 1$

- Horizon (i.e., # of time steps): h

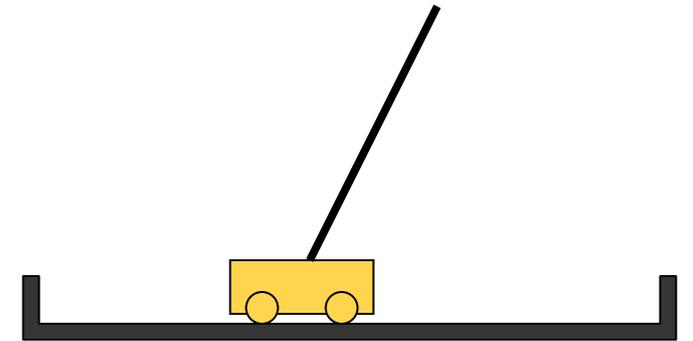
} unknown

- Goal: find optimal policy $\pi^* = \operatorname{argmax}_{\pi} \sum_{t=0}^h \gamma^t E_{\pi}[r_t]$

Example: Inverted Pendulum

- State: $x(t), x'(t), \theta(t), \theta'(t)$
- Action: Force F
- Reward: 1 for any step where pole balanced

Problem: Find $\pi: S \rightarrow A$ that maximizes rewards



Sample Industrial Use Cases

Less Complex

More Complex

Contextual Bandits

Marketing

ad placement,
recommender systems

Loyalty programs

personalized offers

Price management

airline seat pricing
cargo shipment pricing
food pricing

Optimal design

interface personalization

Bayesian Optimization

Hyperparameter optimization

Troubleshooting

Customer assistance

Diagnostics

Fault detection

Design of experiments

Drug design
Material design

Sequential decision Making

Automated trading

Stocks, energy

Optimization

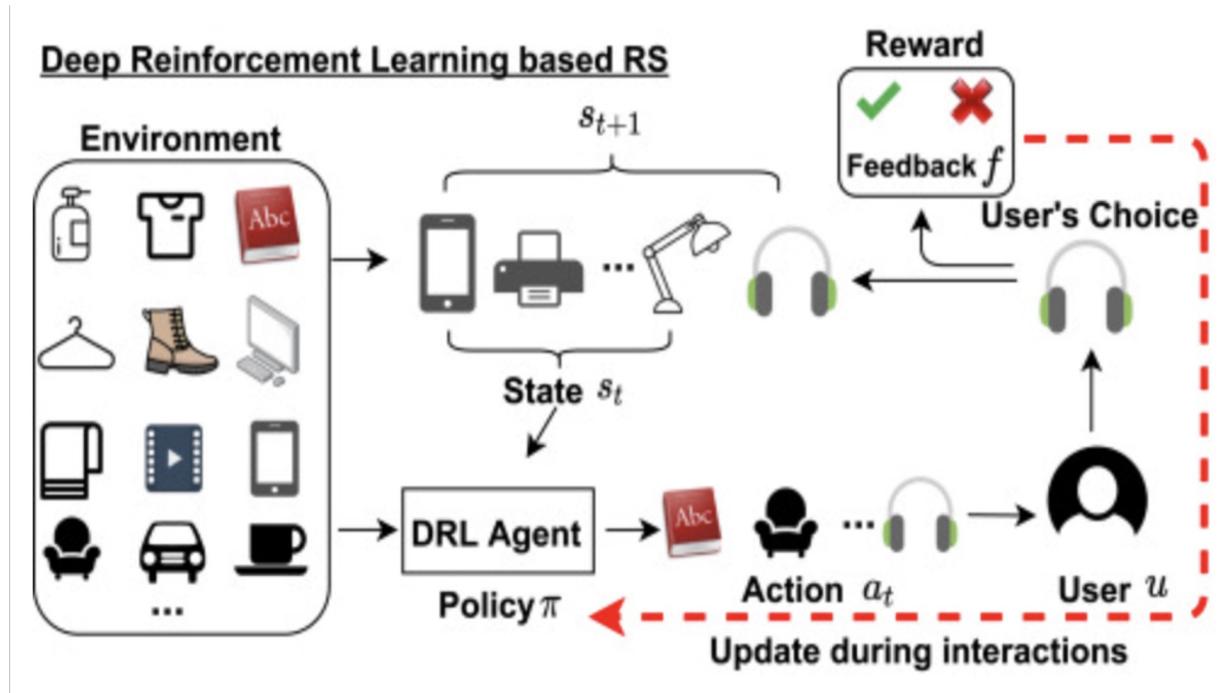
Path planning
Routing
Energy consumption

Control

Robotics
Autonomous driving

Marketing (Recommender System)

- **Agent:** recommender system
- **Environment:** user
- **State:** context, past recommendations and feedback
- **Action:** recommended item
- **Reward:** value of user feedback



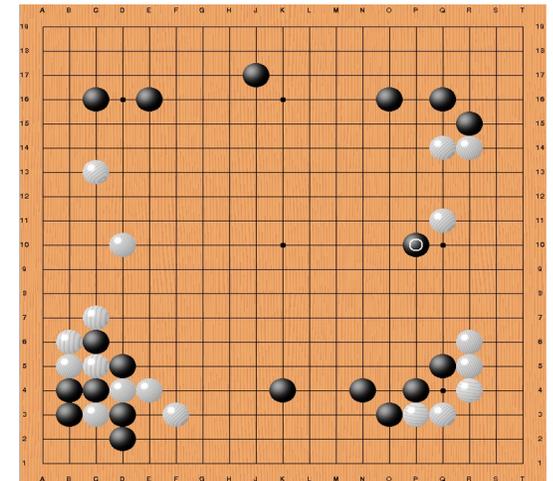
Operations Research (vehicle routing)

- **Agent:** vehicle routing system
- **Environment:** stochastic demand
- **State:** vehicle location, capacity and depot requests
- **Action:** vehicle route
- **Reward:** - travel costs



Game Playing (Computer Go)

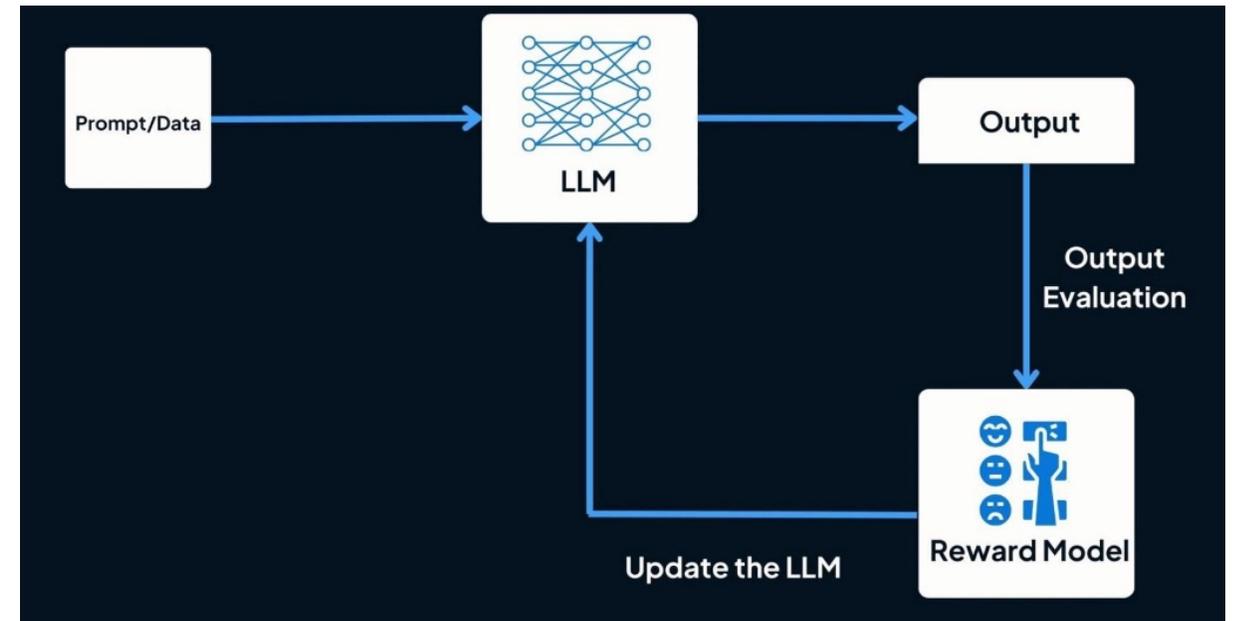
- **Agent:** player
- **Environment:** opponent
- **State:** board configuration
- **Action:** next stone location
- **Reward:** +1 win / -1 loose



- 2016: AlphaGo defeats Lee Sedol (4-1)
 - Game 2 move 37: AlphaGo plays unexpected move (odds 1/10,000)

Conversational Agents (RL from Human Feedback)

- **Agent:** system
- **Environment:** user
- **State:** history of past utterances
- **Action:** system utterance
- **Reward:** task completion, human feedback



Credit: <https://www.twine.net/blog/what-is-reinforcement-learning-from-human-feedback-rlhf-and-how-does-it-work/>

Computational Finance (Trading)

- **Agent:** trading software
- **Environment:** other traders
- **State:** price history
- **Action:** buy/sell/hold
- **Reward:** amount of profit



Example: how to purchase a large # of shares in a short period of time without affecting the price

Important Components in RL

RL agents may or may not estimate the following components:

- **Model:** $\Pr(s' | s, a)$, $\Pr(r | s, a)$
 - Environment dynamics and rewards
- **Policy:** $\pi(s)$
 - Agent action choices
- **Value function:** $V(s)$
 - Expected total rewards of the agent policy

Categorizing RL agents

Value based

- No policy (implicit)
- Value function

Policy based

- Policy
- No value function

Actor critic

- Policy
- Value function

Model based

- Transition and reward model

Model free

- No transition and no reward model (implicit)

Online RL

- Learn by interacting with environment

Offline RL

- No environment
- Learn only from saved data

Bellman's Equation

- Value Iteration:

$$V_n^*(s) \leftarrow \max_a E[r|s, a] + \gamma \sum_{s'} Pr(s'|s, a) V_{n-1}^*(s')$$

- Bellman Equation (when $n \rightarrow \infty$):

$$V^*(s) = \max_a E[r|s, a] + \gamma \sum_{s'} Pr(s'|s, a) V^*(s')$$

- State-action Bellman Equation:

$$Q^*(s, a) = E[r|s, a] + \gamma \sum_{s'} Pr(s'|s, a) \max_{a'} Q^*(s', a')$$

$$\text{where } V^*(s) = \max_a Q^*(s, a), \quad \pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$$

Temporal Difference Control

- Approximate Q-function:

$$Q^*(s, a) = E[r|s, a] + \gamma \sum_{s'} \Pr(s'|s, a) \max_{a'} Q^*(s', a')$$
$$\approx r + \gamma \max_{a'} Q^*(s', a') \longleftarrow \text{one sample approximation}$$

- Incremental update

$$Q_n^*(s, a) \leftarrow Q_{n-1}^*(s, a) + \alpha_n \left(r + \gamma \max_{a'} Q_{n-1}^*(s', a') - Q_{n-1}^*(s, a) \right)$$

learning rate

Tabular Q-Learning

Qlearning()

Initialize s and Q^* arbitrarily

Repeat

Select and execute a

Observe s' and r

Update counts: $n(s, a) \leftarrow n(s, a) + 1$

Learning rate: $\alpha \leftarrow 1/n(s, a)$

$Q^*(s, a) \leftarrow Q^*(s, a) + \alpha \left(r + \gamma \max_{a'} Q^*(s', a') - Q^*(s, a) \right)$

$s \leftarrow s'$

Until convergence of Q^*

Return Q^*

Q-learning Exercise

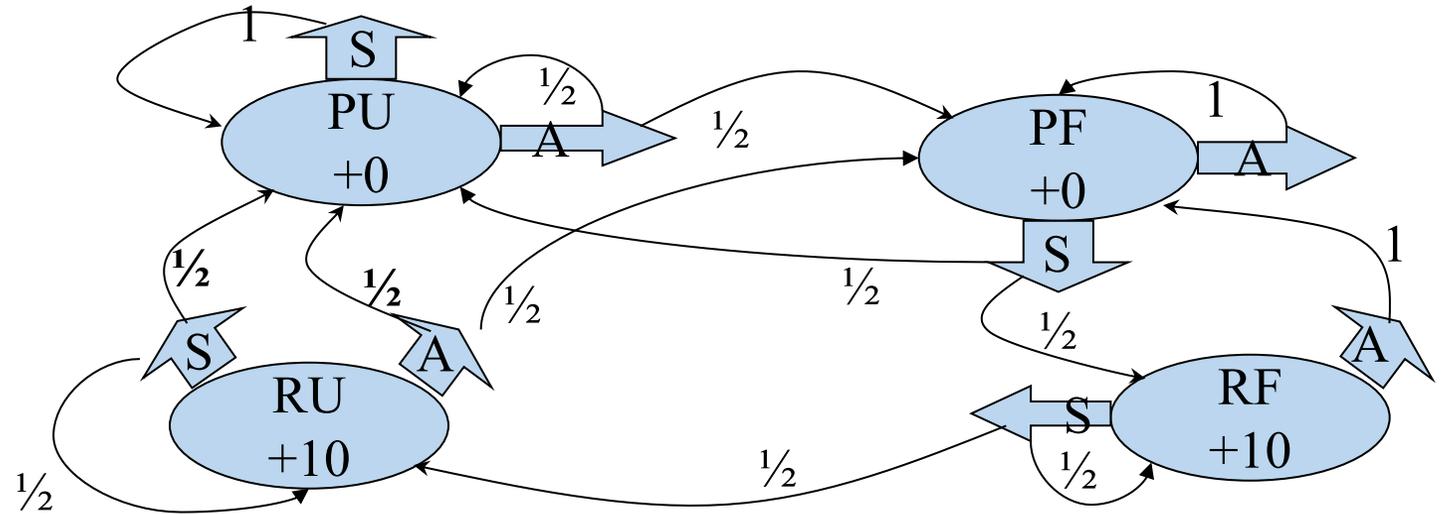
Current estimates:

$$Q(RF, S) = 25$$

$$Q(RF, A) = 20$$

$$Q(RU, S) = 20$$

$$Q(RU, A) = 15$$



Discount: $\gamma = 0.9$

Learning rate: $\alpha = 0.5$

Update $Q(RF, S)$ after executing S in RF and transitioning to RU :

Convergence

- Q-learning converges to optimal Q-values if
 - Every state is visited infinitely often (due to **exploration**)
 - The **action selection becomes greedy** as time approaches infinity
 - The learning rate α is decreased fast enough, but not too fast (sufficient conditions for α):

$$(1) \sum_t \alpha_t \rightarrow \infty \quad (2) \sum_t (\alpha_t)^2 < \infty$$

- NB: $\alpha_t(s, a) = 1/n_t(s, a)$ satisfies the above conditions

Common Exploration Methods

- ϵ -greedy:
 - With probability ϵ , execute random action
 - Otherwise execute best action $a^* = \operatorname{argmax}_a Q(s, a)$

- Boltzmann exploration

- Increasing temperature T increases stochasticity

$$\Pr(a) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_a e^{\frac{Q(s,a)}{T}}}$$